AD-A054 920 AEROSPACE MEDICAL RESEARCH LAB WRIGHT-PATTERSON AFB OHIO F/G 12/1
AN ALGORITHM FOR COMPUTING MATRIX SQUARE ROOTS WITH APPLICATION--ETC(U)
1977 D W REPPERGER
UNCLASSIFIED AMRL-TR-77-21 NL

1 OF 1
AD
A054920

END
DATE
FILMED
7-78
DDC

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

2

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER (14) AMRL-TR-77-21	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) (6) AN ALGORITHM FOR COMPUTING MATRIX SQUARE ROOTS WITH APPLICATION TO THE Riccati Equation Implementation	5. TYPE OF REPORT & PERIOD COVERED
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) D.W. Repperger	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Aerospace Medical Research Laboratory, Aerospace Medical Division, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (17) Ø4 (16) 6893-04-34-62202F
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE (11) 1977
	13. NUMBER OF PAGES 10 (12) 11p.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassfied
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Oral presentation at the 1977 IEEE Conference on Decision and Control, Fairmont Hotel, New Orleans, 7-9 December 1977

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Human operator identification
Riccatti type equations
Square root matrix

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An iterative algorithm is presented for obtaining a positive definite symmetric square root of a positive definite symmetric matrix. This algorithm has application in the implementation of Riccati type equations. The approach presented here has the advantage that apriori upper and lower bounds to the converged answer can be obtained sequentially at any point in the iterative process. These apriori bounds can also be obtained for the Riccati equation using a discrete implementation procedure. Theorems on convergence are proved and examples are worked.

2

FA6 - 10:00

An Algorithm For Computing Matrix Square Roots With Application To Riccati Equation Implementation

D. W. Repperger

Aerospace Medical Research Laboratory
Wright-Patterson Air Force Base, Ohio 45433

Abstract

An iterative algorithm is presented for obtaining a positive definite symmetric square root of a positive definite symmetric matrix. This algorithm has application in the implementation of Riccati type equations. The approach presented here has the advantage that apriori upper and lower bounds to the converged answer can be obtained sequentially at any point in the iterative process. These apriori bounds can also be obtained for the Riccati equation using a discrete implementation procedure. Theorems on convergence are proved and examples are worked.

I. Introduction

The study of optimal control and estimation theory has been influenced in recent years by factorization methods in the analytic expression of the filtering equations [1,2,3]. Computationally the solution of filtering and smoothing equations appears numerically better behaved [4] when the equations are of the square root type in lieu of the standard forms. In the study of Riccati type equations [5], an attempt has been made to apply the factorization methods directly to the steady state Riccati equation but an algorithm to produce these results is required.

In the study of Riccati type equations, many methods exist to determine solutions such as iterative procedures [6,7], the partitioned algorithm approach [8,9,10], the Matrix Sign Function method [11,12], and other methods applicable in numerical integration, e.g. [15]. The Matrix Sign Function approach is one method which has the advantage of obtaining additional non-positive definite solutions through the use of a Symplectic matrix composed of the matrices in the Riccati equation. This aspect has been discussed by Bucy [13] and Potter [14].

This paper will develop an algorithm for determining the square root of a positive definite symmetric matrix. The convergence of the algorithm will be proved with upper and lower apriori bounds on the converged value determined at any point along the sequential process. This algorithm can then be applied to the implementation of Riccati type equations. The Riccati type equations that are applicable for this approach include, but are not limited to, the discrete version of the partitioned Riccati solutions [8,10], and the various covariance expressions which occur in square root filtering methods already discussed [3,15]. It is noted that the square root algorithm presented here will be compared (through examples) to the Matrix Sign approach [11] and not to the triangular factorization methods. In a hard to find reference [16] another (but different) algorithm was developed which has been discussed by Bellman [17]. The algorithm in [16], however, has limitations on the norm of the matrix considered and appears to be related to a version of the spectral factorization problem. In Astrom's book [18], similar algorithms result from studies of related spectral factorization problems.

In order to better understand why the factorization methods yield algorithms which have numerical advantages over other methods, this paper is divided in four parts. First the motivation for using this approach is demonstrated by working a scalar example using an algorithm from Number Theory called Euclid's algorithm to characterize an irrational number in terms of continued fractions . Such methods are used in Number Theory [19,20], and for scalar irrationals (such as a square root), they give rise to definitions such as "most efficient expansion" and "best possible approximation". By defining a structure of the continued fraction expansion subject to certain constraints, the definition of "most efficient expansion" can be given in an explicit manner. Using a sequential procedure defined as Euclid's algorithm [19,20], the continued fraction expansion is obtained for the irrational number π . Euclid's method (or a "most efficient expansion") is then constructed for a scalar square root. Apriori upper and lower bounds are obtained for both expansions of these two scalar irrational numbers.

Part II of this paper introduces the "square root algorithm" which differs from Euclid's method. The square root algorithm is applied first to the scalar square root and the resulting continued fraction expansion is compared to Euclid's method. The apriori upper and lower bounds are also calculated. The third part of this paper extends the scalar square root algorithm to the matrix case. Theorems on convergence are proved and examples are worked. The matrix square

*The research reported in this paper was sponsored by Aerospace Medical Research Laboratory, Aerospace Medical Division, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio 45433. Further reproduction is authorized to satisfy needs of the U.S. Government.

root algorithm considered here is compared (through an example) to the Matrix Sign Approach and the convergence of the algorithm is studied numerically.

Finally, the fourth part of this paper considers the application of this approach to the implementation of Riccati type equations. A manner of obtaining apriori upper and lower bounds is demonstrated for Riccati type equations. Some theorems are given and an algorithm is demonstrated which extends the results obtained here for the matrix case into an algorithm for computing the Riccati equation. The apriori upper and lower bounds for the Riccati equation are obtained sequentially as in the matrix case. First, part I of this paper will introduce Euclid's algorithm and provide a motivation for using factorization algorithms.

II. Part-I Euclid's Algorithm-A Motivation For Using Square Root Methods

It is well known that numerical problems occur in computing Riccati solutions or other matrix equations if the matrices are ill conditioned [21]. One measure of ill conditioning occurs if the eigenvalues of the free system are seperated by more than one order of magnitude. One would expect, therefore, that factorization methods should prove to have numerical advantages because the square root matrix may have eigenvalues with less dispersion then the original free system matrix. In an effort to study this effect numerically, Euclid's algorithm is introduced and a scalar example is worked from Number Theory on an irrational number. The purpose of this example is to illustrate to the reader the relationship between the factorization methods to continued fraction approaches and also to define the numerical advantages (in terms of a definition called "most efficient") of the factorization methods considered here. First some definitions are necessary.

Let ξ_o denote a scalar irrational number. The problem of interest is to determine a continued fraction expansion of ξ_o in terms of rational numbers. First the definition of a continued fraction is specified as follows [19,20]:

<u>Define</u>:

$$W = \langle a_o, a_1, \cdots, a_N, \cdots \rangle \quad (1)$$

which is notation for:

$$W = a_o + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cdots + \cfrac{1}{a_N + \cdots}}}} \quad (2)$$

where each a_i is subject to the constraint of being positive and an integer. Then Euclid's algorithm [20] is defined as follows:

Denote: $$a_i = [\xi_i] \quad (3)$$

where a_i is defined as the nearest integer smaller than the irrational number ξ_i. Euclid's algorithm for computing the a_i from the ξ_i proceeds as follows:

$a_o = [\xi_o]$, now proceeding as an algorithm, let,

$$\xi_1 = \frac{1}{\xi_o - a_o}$$

and let $a_1 = [\xi_1]$. Inductively it follows that:

$$\xi_{i+1} = \frac{1}{\xi_i - a_i}$$

and $a_i = [\xi_i]$ and this completes the algorithm for all a_i. It is now appropriate to give a definition of "most efficient expansion" as it relates in the context of the continued fractions in equations (1,2).

<u>Definition</u>: "Most Efficient Expansion"

The most efficient [19,20] expansion of an irrational scalar number ξ_o by a continued fraction representation is the one in which ξ_o can be accurately expressed to as many decimals as possible by the fewest number of integer terms in the expansion. It is noted that W is subject to the constraint of equation (2) with each a_i positive integers. It has been shown [19,20] that Euclid's method satisfies this property for scalar irrationals. An example will now be worked with the irrational number π to illustrate Euclid's algorithm.

<u>Example (1)</u>:

Let ξ_o be the irrational number π, i.e. $\xi_o = 3.14159+$ Hence $a_o = [\xi_o] = 3$. To calculate a_1, compute

$$\xi_1 = \frac{1}{\xi_o - a_o}$$

or

$$\xi_1 = \frac{1}{\xi_o - 3} = \frac{1}{.14159} = 7.06+$$

Now compute $a_1 = [\xi_1] = 7$. In this manner we calculate (using double precision (29 digits) on a CDC computer) the following numerical results:

$$\begin{aligned}
\pi = \xi_o &= 3.1415926535897932384626433832 79\\
\xi_1 &= 7.0625133059310457697930051531\\
\xi_2 &= 15.996594406685694113105998 74\\
\xi_3 &= 1.0034172310133977856369454934\\
\xi_4 &= 292.63459101223866070785852178\\
\xi_5 &= 1.5758180949841629954527421461\\
\xi_6 &= 1.7366595609113341887024369765\\
\xi_7 &= 1.3574791573503535151332712575\\
\xi_8 &= 2.7973658867611542733590927681\\
\xi_9 &= 1.2541293985649846777332901666\\
\xi_{10} &= 3.9350032135077287460064498245\\
\xi_{11} &= 1.0695150407541735890218572566\\
\xi_{12} &= 14.385376015764735321479897720
\end{aligned}$$

This gives rise to $a_o=3$, $a_1=7$, $a_2=15$, $a_3=1$, $a_4=292$, $a_5=1$, $a_6=1$, $a_7=1$, $a_8=2$, $a_9=1$, $a_{10}=3$, $a_{11}=1$, and $a_{12}=14$ which results in the following partial fraction expansion:

$$\pi = 3 + \cfrac{1}{7 + \cfrac{1}{15 + \cfrac{1}{1 + \cfrac{1}{292 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{3 + \cfrac{1}{1 + \cfrac{1}{14 + \cdots}}}}}}}}}}}} \quad (4)$$

In order to study convergence and to develop apriori bounds, it is of interest to study the partial sums of equation (4). Let $r_o=a_o$, $r_1= a_o + 1/a_1$, and in general:

$$r_N = \langle a_o,\ a_1,\ \cdots,\ a_N \rangle \qquad (5)$$

It has been shown explicitly [19,20] that for the scalar case, the following upper and lower bounds exist for π :

$$r_o<r_2<r_4<\cdots< \lim_{N\to\infty} r_N = \pi < \cdots<r_5<r_3<r_1 \qquad (6)$$

which gives rise to apriori bounds of the form:

$$3 < \frac{333}{106} < \cdots < \pi < \cdots < \frac{355}{113} < \frac{22}{7} \qquad (7)$$

The scalar version of this proof was presented for the square root of a scalar number in [22], the matrix case will be presented in the sequel. Therefore by calculating r_o and r_1, an upper and lower bound on π can be determined apriori. By then calculating r_2 and r_3, an even more accurate bound on π can then be determined. This procedure can be continued indefinitely with convergence to values as close as desired. To illustrate to the reader how these apriori bounds can be obtained for the irrational number π, the terms $r_o, \cdots, r_8$ are calculated for the fraction in equation (4).

r_o	=	3	= 3.0
r_1	=	22/7	= 3.142857142857+
r_2	=	333/106	= 3.14150943396226415094+
r_3	=	355/113	= 3.14159292035+
r_4	=	103993/33102	= 3.141592653011+
r_5	=	104348/33215	= 3.14159265392142+
r_6	=	208341/66317	= 3.141592653467+
r_7	=	312689/99532	= 3.1415926536189+
r_8	=	833719/265381	= 3.14159265358107777+

It is easily seen from these numerical values that the r_i satisfy the relationships specified by equation (6). Also from the example it is observed that the first four terms of the expression r_4 give accuracy beyond 7 decimal places. One should now be motivated to apply Euclid's algorithm to a quadratic equation to observe the results.

III Euclid's Algorithm Applied To A Scalar Square Root

Taking as an example the square root of 3, the most efficient method will be studied numerically using example (2):

Example (2):

Expanding the square root of 3 to 30 decimal places yields:

$\xi_o = \sqrt{3} = 1.732050807568877293527446341505$. Now the calculation of the a_i and ξ_i will proceed as for π:

$$a_o = [\ 1.732+\ ] = 1$$

$$\xi_1 = \frac{1}{\sqrt{3} - 1} = 1.36+$$

Hence $a_1 = [\xi_1] = 1$, A summary of the first 11 terms yields:

ξ_1	=	1.3660254037844363479757026605
ξ_2	=	2.7320508075688992137673133780
ξ_3	=	1.3660254037843965712518405766
ξ_4	=	2.7320508075691909097423020388
ξ_5	=	1.3660254037838529560257256617
ξ_6	=	2.7320508075732481355762416877
ξ_7	=	1.3660254037762821195840238963
ξ_8	=	2.7320508076297576012776604042
ξ_9	=	1.3660254036708340246350351294
ξ_{10}	=	2.7320508084168328955065207969
ξ_{11}	=	1.3660254022021315334827392008

which results in $a_o=1, a_1=1, a_2=2, a_3=1, a_4=2, a_5=1, a_6=2, a_7=1, a_8=2, a_9=1$, and $a_{10}=2$. In this manner the "most efficient" method to calculate the square root of 3 is as follows using the continued fraction approach:

$$3 = 1 + \cfrac{1}{1 + \cfrac{1}{2+ \cfrac{1}{1+ \cfrac{1}{2+ \cfrac{1}{1+ \cfrac{1}{2+ \cdots}}}}}} \qquad (8)$$

The periodic nature of the a_i terms in equation (8) is of interest because it is known that in Riccati equation expansions of the partitioned algorithm approach [8], this same periodicity occurs. The periodicity obtained here numerically occurs due to:

$$\xi_j = \xi_{j+2} \quad \text{for all } j$$

$$\text{with} \quad a_i = 1 \quad (i \text{ odd}) \qquad a_{i+1} = 2 \quad (i \text{ odd})$$

and this is a result due to the following identity:

$$1 + \sqrt{3} = \cfrac{1}{\cfrac{1}{\sqrt{3} - 1} - 1}$$

The results obtained here so far indicate that using the "most efficient" approach, the square root methods have a periodic property. If this method were to be applied to the matrix case, the results do not extend readily due to the fact that the definition "most efficient" does not have the same meaning in the matrix case. In order to have an algorithm that does extend to the matrix case, the square root algorithm will next be presented. The square root algorithm gives identical results to the "most efficient" algorithm for the scalar case. The square root algorithm, however, can be extended to obtain matrix square roots, to implement the Riccati equation, and , in addition, to determine upper and lower apriori bounds.

IV - Part II- The Square Root Algorithm (Scalar Case)

Consider example (2) to determine the square root of 3 via the following sequence of steps:

$$\text{Let} \quad W = \sqrt{3}$$

Rewrite this as

$$W = 1 + (\sqrt{3} - 1) \qquad (9)$$

$$\text{or} \quad W = 1 + \cfrac{1}{\frac{1}{2}(\sqrt{3} + 1)} = 1 + \cfrac{1}{\frac{1}{2} + \frac{1}{2}(1+\sqrt{3} - 1)}$$

$$W = 1 + \cfrac{1}{1+\frac{1}{2}(\sqrt{3}-1)} = 1 + \cfrac{1}{1+\frac{1}{2}\left[\cfrac{1}{\frac{1}{2}(\sqrt{3}+1)}\right]} \qquad (10)$$

or:

$$W = 1 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{2+ \ddots}}}}}} \qquad (11)$$

Since 3 = 1.7320508+ , then the continued fraction of interest is:

$$1.732+ = 1 + \cfrac{1}{1+ \cfrac{1}{2+ \cfrac{1}{1+ \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{2+ \ddots}}}}}} \qquad (12)$$

This square root approach yields numerical answers for this example which are identical to the "most efficient" method. For completeness the partial sums r_o, $r_1, \cdots, r_{10}$ are computed and displayed here:

$$\begin{aligned}
r_o &= 1 \\
r_1 &= 2/1 = 2.0 \\
r_2 &= 5/3 = 1.667 \\
r_3 &= 7/4 = 1.750 \\
r_4 &= 19/11 = 1.7272 \\
r_5 &= 26/15 = 1.7333 \\
r_6 &= 71/41 = 1.7317 \\
r_7 &= 97/56 = 1.73214 \\
r_8 &= 265/153 = 1.732026 \\
r_9 &= 362/209 = 1.732057 \\
r_{10} &= 989/571 = 1.732049
\end{aligned}$$

It is easily demonstrated that:

$$r_o < r_2 < r_4 < r_6 < r_8 < r_{10} < \cdots \sqrt{3} < \cdots < r_9 < r_7 < r_5 < r_3 < r_1 \qquad (13)$$

and approximations to the square root of 3 can be determined by calculating only r_o and r_1; or more accurately be calculating r_2 and r_3, etc. This approach will now be extended to the matrix case.

V-Part III-Extension of The Algorithm For Determining Matrix Square Roots

In order to develop an algorithm for matrix square roots, it is necessary to accurately define the matrices of interest. Let S_1 be an (nxn) positive definite symmetric matrix. The (nxn) matrix S_o which is the square root of S_1 is required to be positive definite symmetric and satisfy:

$$S_o S_o^T = S_1 \qquad (14)$$

where T denotes matrix transpose. S_o is defined as the the square root matrix of S_1. For simplicity in the ensuing derivation, it will be assumed that:

$$S_o - I > 0 \quad \text{and} \quad S_1 - I > 0 \qquad (15)$$

where the inequality representation for two matrices $A > B$ implies that A-B is positive definite and I is the nxn identity matrix. If equation (15) is not satisfied for a specified S_o and S_1, then we require:

$$S_o - \varepsilon I > 0, \quad \text{and} \quad S_1 - \varepsilon^2 I > 0 \qquad (16)$$

for some $\varepsilon > 0$. The ensuing derivation follows exactly with (16) replaced by (15) , but for notation simplicity, the inequality of equation (15) will be used. To derive the square root algorithm, proceed as in equations (9-12) for the scalar case.

Denote:

$$S_o = S_1^{1/2} \qquad (17)$$

This can be written as:

$$S_o = I + S_1^{1/2} - I \qquad (18)$$

Theorem 1 will now be helpful in the derivation:

Theorem 1:

$$S_1^{1/2} - I = (S_1-I)(S_1^{1/2}+I)^{-1} = (S_1^{1/2}+I)^{-1}(S_1-I) \qquad (19a)$$

$$S_1^{1/2} + I = (S_1-I)(S_1^{1/2}-I)^{-1} = (S_1^{1/2}-I)^{-1}(S_1-I) \qquad (19b)$$

Proof: Consider the identities:

$$(S_1^{1/2}-I)(S_1^{1/2}+I) = S_1-I \qquad (20a)$$

$$(S_1^{1/2}+I)(S_1^{1/2}-I) = S_1-I \qquad (20b)$$

Equation (19a) follows by pre and post multiplying equations (20a-b) by $(S_1^{1/2}+I)^{-1}$ which is positive definite. Equation (19b) follows by pre and post multiplying equations (20a-b) by $(S_1^{1/2}-I)^{-1}$ which is known to be positive definite by the inequality (15).

Q.E.D.

The derivation of the algorithm will now continue by rewriting equation (18) as follows:

$$S_o = I + (S_1^{1/2}-I)(S_1^{1/2}+I)(S_1^{1/2}+I)^{-1} \qquad (21)$$

From equation (20a) this implies (21) can be written:

$$S_o = I + (S_1-I)\,[S_1^{1/2}+I]^{-1} \qquad (22)$$

Since for two matrices A,B, $(AB)^{-1}=B^{-1}A^{-1}$, it is desired to represent equation (22) as:

$$S_o = I + I\,[\,(S_1^{1/2}+I)(S_1-I)^{-1}\,]^{-1} \qquad (23)$$

This can be written as:

$$S_o = I + I\,[\,(S_1-I)^{-1} + S_1^{1/2}(S_1-I)^{-1}\,]^{-1} \qquad (24)$$

Now the procedure used in equations (17-18) will be reapplied to equation (24) resulting in:

$$S_o = I + I\,[\,(S_1-I)^{-1} + [I+S_1^{1/2}-I](S_1-I)^{-1}\,]^{-1} \qquad (25)$$

It is worthwhile to rewrite this as:

$$S_o = I + I\,[2(S_1-I)^{-1} + [\,S_1^{1/2}-I](S_1-I)^{-1}\,]^{-1}$$

Using the results of equation (19b) yields:

$$S_o = I + I[2(S_1-I)^{-1} + [S_1^{1/2}+I]^{-1}\,]^{-1} \qquad (26)$$

Now expanding the inner term yields:

$$S_o=I+I[\ 2(S_1-I)^{-1} +I[\ I+(S_1^{1/2} +I-I)]^{-1}\ ]^{-1} \qquad (27)$$

This can be rewritten as:

$$S_o = I + I[\ 2(S_1-I)^{-1} + I[2I+(S_1^{1/2}-I)]^{-1}\ ]^{-1} \qquad (28)$$

The observant reader may now compare the term $(S_1^{1/2}-I)$ in equation (18) with the same term in the inner most brackets of equation (28). The next step in the derivation is to repeat the procedure from equations (18-28) resulting in the next expression as follows:

$$S_o=I+I[2(S_1-I)^{-1}+I[2I+(S_1^{1/2}-I)(S_1^{1/2}+I)(S_1^{1/2}+I)^{-1}]^{-1}]^{-1} \qquad (29)$$

etc. Since the derivation is periodic from this point on, the results can be summarized as follows:

$$\text{Let} \quad S_A = 2I \qquad (30a)$$

$$S_B = 2(S_1-I)^{-1} \qquad (30b)$$

Then:

$$S_o = I+I[S_B+I[S_A+I[S_B+I[S_A+I[S_B\cdots]^{-1}]^{-1}]^{-1}]^{-1} \qquad (31)$$

This could also be written as follows:

$$S_o = I + \cfrac{I}{S_B + \cfrac{I}{S_A + \cfrac{I}{S_B + \cfrac{I}{S_A + \cfrac{I}{S_B + \cdots}}}}} \qquad (32)$$

The expressions (31-32) obtained are desired for several reasons. First, the matrix case reduces to the "most efficient" approach as S_A and S_B become scalars. Secondly, the matrices S_A and S_B are positive definite and it is desireable to have this property to obtain apriori bounds. Thirdly, it appears that this method has application in the implementation of the Riccati equation as observed in references [3,8,15]. The theorems on convergence will now be given with their proofs in Appendix (A). Some examples will then be worked to illustrate this approach.

VI-Convergence Proofs of The Square Root Algorithm

In order to establish a theorem on the convergence of the nested matrix sequence (31), a sequence of partial sums of the expression (31) will be derived. Denote R_i, $i=0,1,\cdots$ as the matrix partial sums of (31) which reduce to the scalar partial sums r_i defined in equation (5).

Let:

$$R_o = I \qquad (33a)$$

$$R_1 = I + I[\ 2(S_1-I)^{-1}]^{-1} \qquad (33b)$$

$$R_2 = I + I[2(S_1-I)^{-1} +I[2I]^{-1}]^{-1} \qquad (33c)$$

$$R_3 = I+I[2(S_1-I)^{-1}+I[2I+[2(S_1-I)^{-1}]^{-1}]^{-1}]^{-1} \qquad (33d)$$

or as an algorithm:

$$R_o = I \qquad (34a)$$

$$R_1 = I + I[2(S_1-I)^{-1}]^{-1} \qquad (34b)$$

$$\vdots$$

$$R_N = I +[2(S_1-I)^{-1}+I[I+R_{N-2}]^{-1}]^{-1} \qquad (34c)$$

Theorem 2 contains the properities of convergence of this algorithm.

<u>Theorem 2:</u>

$$\lim_{N\to\infty} R_N = S_1^{1/2} \qquad (35a)$$

with the following apriori bounds:

$$R_o<R_2<R_4<R_6<\cdots<R_N = S_1^{1/2}< \cdots<R_7<R_5<R_3<R_1$$

$$\lim N\to\infty \qquad (35b)$$

The proof of theorem (2) is given in appendix (A). The scalar version of this proof was given in [22] and similar proofs of the scalar example are given in the references on Number Theory [19,20]. Some examples will now be worked to illustrate the convergence properities of the algorithm and also how the apriori bounds may be used.

VII Some Numerical Examples Using The Matrix Square Root Algorithm:

As the simplist example to observe the procedure presented here, consider the square root of the following matrix:

<u>Example 3</u>

$$S_1 = \begin{bmatrix} 4.0\ , & 0.0 \\ 0.0\ , & 25.0 \end{bmatrix}$$

Using the algorithm results in R_{10} and R_{11} as follows:

$$R_{10}=\begin{bmatrix} 1.999797, & 0.0 \\ 0.0 & ,\ 4.746472 \end{bmatrix} \text{ and } R_{11}=\begin{bmatrix} 2.000068, & 0.0 \\ 0.0\ , & 5.176476 \end{bmatrix}$$

Please recall the inequality:

$$R_N < S_1^{1/2} < R_M$$

is true for all N even integers and all M odd integers. Also, the apriori bounds R_{10} and R_{11} indicate the accuracy of this procedure.

<u>Example 4</u>

$$S_1 = \begin{bmatrix} 4.0 & , & 6.0 \\ 6.0 & , & 25.0 \end{bmatrix} \qquad (36)$$

Using the Matrix Square Root Algorithm results in R_{30} and R_{31} as follows:

$$R_{30}=\begin{bmatrix}1.788847 , & .894398\\ .894398 , & 4.919241\end{bmatrix},\quad R_{31}=\begin{bmatrix}1.788860 , & .894447\\ .894447 , & 4.919423\end{bmatrix}$$

It is interesting to compute the products $R_{30}R_{30}^T$ and $R_{31}R_{31}^T$.

$$R_{30}R_{30}^T = \begin{bmatrix}3.999921371 , & 5.999700491\\ 5.999700491 , & 24.9988798\end{bmatrix},$$

$$R_{31}R_{31}^T = \begin{bmatrix}4.0000555, & 6.0002036\\ 6.0002036, & 25.00075808\end{bmatrix}$$

which can be compared to S_1 of equation (36). The next example was taken from reference [11] to serve as a comparison to the approach presented here and also to look at higher order systems.

Example 5

$$S_1 = \begin{bmatrix}1.2 \times 10^5 , & 230. , & 10.\\ 230. & , 1000. , & 1.\\ 10. , & 1. & , 0.5\end{bmatrix}$$

Using the Matrix Sign Approach [11], Denman found the computed square root as:

$$S_o = \begin{bmatrix}346.40961 , & 0.608420 , & .0287555\\ 0.608420 , & 31.61690 , & .0303966\\ 0.0287555 , & 0.0303966, & 0.705876\end{bmatrix}$$

Using the Matrix Square Root Algorithm presented here, the results obtained were:

$$R_{632}= \begin{bmatrix}328.739943 , & .574269 , & .027283\\ .574269 & , 31.616842, & .030394\\ .027283 & , .030394 & , .705868\end{bmatrix}$$

$$R_{633}= \begin{bmatrix}364.918997 , & .644195, & .030298\\ .644195 & , 31.616978, & .030400\\ .030298 & , .030400 & , .705868\end{bmatrix}$$

Obviously for higher order systems, some modes converge much faster than other modes. In other words the ill conditioning of S_1 will effect this algorithm (by slowing down its convergence) in a manner dependent on the degree of ill conditioning of S_1. Also the effects of numerical bias occur in this algorithm as can be seen because the element in the third row, third column of R_{632} and R_{633} has apparently converged independent of the remaining elements. Continuing the calculation of R_N for $N > 2000$ gave no additional change in this term. This element may be numerically biased (through roundoff errors or truncation errors involved in the matrix inversion subroutine used here). An excellent discussion of this problem can be found in Bierman [4]. One more example was worked to numerically study S_1 in the event it was only positive semi-definite.

Example 6

$$S_1 = \begin{bmatrix}4.0 , & 10.0\\ 10.0 , & 25.0\end{bmatrix} \tag{37}$$

The algorithm yields:

$$R_{30}=\begin{bmatrix}.772480 , & 1.844994\\ 1.844994, & 4.646968\end{bmatrix},\quad R_{31}=\begin{bmatrix}.771536 , & 1.845506\\ 1.845506, & 4.647099\end{bmatrix}$$

The interesting result occurs by considering the products $R_{30}R_{30}^T$ and $R_{31}R_{31}^T$.

$$R_{30}R_{30}^T = \begin{bmatrix}4.00072821 , & 9.998849043\\ 9.998849043 , & 24.998314450\end{bmatrix}$$

$$R_{31}R_{31}^T = \begin{bmatrix}4.001202784 , & 10.000123404\\ 10.000123404 , & 25.001421516\end{bmatrix}$$

which can be compared to S_1 of equation (37); this procedure appears to work if S_1 is only positive semi-definite. Theorem (3) illustrates that a best estimate of a matrix square root can be obtained by taking the mean value of the best two apriori bounds.

Theorem 3:

(3a) The "best estimate" of a matrix square root can be obtained as follows:

$$\hat{S}_o = \max_N \; (1/2)\;[\,R_N + R_{N+1}] \tag{38a}$$

(3b) The error in the estimate specified by equation (38a) can be no worse than the following bound:

$$\|S_o - \hat{S}_o\| \leq \|R_{N+1} - R_N\| \tag{38b}$$

Hence even if S_O is not known, R_{N+1} and R_N provide an estimate of the accuracy of this procedure. One may now take appropriate matrix norms and examine examples 3,4, and 5. Appendix B demonstrates a proof of this theorem with an appropriate definition of "best estimate". This result can be seen numerically in examples (3-5). The estimate specified by equation (38a) is as accurate a method as possible to guess a matrix square root. The methods obtained here will now be applied to the discrete version of the Riccati equation.

VIII Part IV-Application of This Approach To Discrete Implementation of The Riccati Equation

In the implementation of the continuous Riccati equation at discrete times $0=t_o<t_1<t_2$ with $\Delta = t_{k+1} - t_k$, the following recursion relationships can be obtained:

$$P_{k+1} = P_0 + A\,[\,W + P_k^{-1}\,]^{-1}\,A^T \tag{39}$$

where P_o, A, and W are quantities obtained over the time interval Δ of a Riccati equation with zero initial conditions. Equation (39) appears in one form or another in the partitioned algorithm approach of Lainiotis [8], the square root filtering algorithms of Morf and Kailath [3], and in Potter and Womble [15]. To apply this approach similar to the manner in which the matrix algorithm (34a-c) was obtained, note the following sequences:

$$\bar{R}_o = P_o$$
$$\bar{R}_1 = P_o + A[W]^{-1}A^T$$
$$\bar{R}_2 = P_o + A[W+[P_o]^{-1}]^{-1}A^T$$
$$\bar{R}_3 = P_o + A[W+[P_o+A[W]^{-1}A^T]^{-1}]^{-1}A^T$$

The Riccati Equation Algorithm becomes:

$$\bar{R}_o = P_o \tag{40a}$$
$$\bar{R}_1 = P_o + A[W]^{-1}A^T \tag{40b}$$
$$\vdots$$
$$\bar{R}_N = P_o + A[W+(\bar{R}_{N-2})^{-1}]^{-1}A^T \tag{40c}$$

Theorem (4) contains the application of the approach here for Riccati equations.

Theorem 4:
If $P_o > 0$, $A > 0$, and $W > 0$, then:

(a) All $\bar{R}_i > 0$, $i=0,1,\cdots$

(b) $\bar{R}_N < P_K < \bar{R}_M$, N even, M odd with $N<K$, and $M<K$.

(c) The following apriori bounds exist as $N\to\infty$ (or as $\Delta t\to 0$) for the continuous case:

$$\bar{R}_o < \bar{R}_2 < \bar{R}_4 < \bar{R}_6 < \cdots < \lim_{\substack{N\to\infty \text{ or} \\ \Delta t\to 0}} P_N < \cdots < \bar{R}_7 < \bar{R}_5 < \bar{R}_3 < \bar{R}_1$$

with apriori bounds:

$$P_o < P_o + A[W+(P_o)^{-1}]^{-1}A^T < \cdots < \lim_{N\to\infty} P_N < \cdots <$$
$$< \cdots < P_o + A[W+(P_o+A[W]^{-1}A^T]^{-1}]^{-1}A^T < P_o + A[W]^{-1}A^T$$

i.e. as many apriori bounds can be found as is desired for the numerical scheme.

(d) The best estimate of P_N (in the continuous case) can be expressed as follows:

$$\hat{P}_N = \max_N \; (1/2)\,[\,\bar{R}_N + \bar{R}_{N+1}] \tag{41}$$

(e)The error in the estimate specified by equation (41) can be no worse than the following bound:

$$\|P_N - \hat{P}_N\| \le \|\bar{R}_{N+1} - \bar{R}_N\|$$

The proof of theorem (4) is outlined in Appendix (C).

The results of this last section seem most interesting in the study of the accuracy of methods to implement the Riccati equation. It is the contention here that since these methods reduce to the "most efficient" method in the scalar case, they should be considered first in the calculation of Riccati type equations.

IX Summary and Conclusions

An iterative algorithm is presented for obtaining matrix square roots. The techniques used to derive the algorithm can be used in discrete implementation of the Riccati equation. The advantage of this approach is the development of apriori bounds on both the matrix computations and the Riccati solutions. Examples are worked to illustrate this method.

Appendix A

In this appendix all inequalities refer to matrix inequalities, i.e. $A>B$ implies $x^T[A-B]x$ is positive *for all vectors x of the appropriate dimensions. To* prove theorem 2, lemmas 1-7 must be shown to be true.

Lemma (1):
Let A_1, B, A_2, C_1, C_2 be positive definite matrices with:

$$A_1 = [\,B^{-1} + C_1^{-1}\,]^{-1} \tag{A.1}$$
$$A_2 = [\,B^{-1} + C_2^{-1}\,]^{-1} \tag{A.2}$$

if $C_1 > C_2$, then $A_1 > A_2$.

Proof:

$$A_1^{-1} = B^{-1} + C_1^{-1}$$
and $$A_2^{-1} = B^{-1} + C_2^{-1}$$

since $C_1 > C_2$ and both matrices are positive definite then we have [17]: $C_2^{-1} > C_1^{-1}$ but

$$C_2^{-1} = A_2^{-1} - B^{-1}$$
$$C_1^{-1} = A_1^{-1} - B^{-1}$$

or $$A_2^{-1} - B^{-1} > A_1^{-1} - B^{-1} \tag{A.3}$$

or $A_2^{-1} > A_1^{-1}$, but since both matrices are positive definite implies $A_1 > A_2$.

Q.E.D.

Lemma 2:
Let A,B, and C be positive definite matrices with $C = (A^{-1} + B^{-1})^{-1}$, then the following inequalities arise:

$$C < B \tag{A.4}$$
$$C < A \tag{A.5}$$

Proof of Lemma 2:

Note $C^{-1} = A^{-1} + B^{-1}$

Hence $C^{-1} > A^{-1}$

And $C^{-1} > B^{-1}$

Therefore $C < B$ and $C < A$ follows.

Q.E.D.

Lemma 3

The matrix subsequences R_N for $N=0,2,4,6,\cdots$ (all even integers) form a monotonically increasing matrix sequence, i.e.

$$R_o < R_2 < R_4 < R_6 < \cdots\cdots < R_N < \cdots\cdots <$$

Proof:

The proof is by mathematical induction. By calculation:

$$R_o = I$$

$$R_2 = I+[2(S_1-I)^{-1}+(2I)^{-1}]^{-1} > I = R_o$$

Hence $R_2 > R_o$. For notation simplicity let $B = \frac{S_1-I}{2} > 0$, then:

$$R_2 = I + [B^{-1} + (I+R_o)^{-1}]^{-1} \qquad (A.6)$$

$$R_4 = I + [B^{-1} + (I+R_2)^{-1}]^{-1} \qquad (A.7)$$

But from lemma (1) if $I+R_2 > I+R_o$, and using (A.6) and (A.7) yields:

$$R_4 > R_2$$

Now assume the results hold for R_N, it is necessary to show that they hold for R_{N+2}. Note we assume

$$R_N > R_{N-2} \qquad \text{(N even)}$$

but

$$R_N = I + [B^{-1} + (I+R_{N-2})^{-1}]^{-1}$$

$$R_{N+2} = I + [B^{-1} + (I+R_N)^{-1}]^{-1}$$

But from lemma (1) if $(I+R_N) > I+R_{N-2}$ implies $R_{N+2} > R_N$. Since the results hold for N=0, and N=2, they hold for all N.

Q.E.D.

Lemma (4):

The matrix subsequences R_N for $N=1,3,5,7,\cdots$ (all odd integers) form a monotonically decreasing matrix sequence, i.e.

$$R_1 > R_3 > R_5 > R_7 > R_9 > \cdots\cdots >$$

Proof

Again the proof is by mathematical induction. By calculation:

$$R_1 = (1/2)\ [S_1 + I]$$

$$R_3 = I + \left[\left(\frac{S_1-I}{2}\right)^{-1} + I[2I+[2(S_1-I)^{-1}]^{-1}]^{-1}\right]^{-1}$$

But using lemma (2) implies:

$$R_3 - I < \frac{S_1 - I}{2}$$

or

$$R_3 < \frac{S_1+I}{2} = R_1$$

Hence

$$R_3 < R_1$$

For notation simplicity let $B = \frac{S_1-I}{2} > 0$, then:

$$R_3 = I + [B^{-1} + (I+R_1)^{-1}]^{-1}$$

$$R_5 = I + [B^{-1} + (I+R_3)^{-1}]^{-1}$$

Now use lemma (1). If $I+R_1 > I+R_3$, then this implies from lemma (1) that $R_3 > R_5$ or $R_1 > R_3 > R_5$. Now assume the results hold for R_N, it is necessary to show that they hold for R_{N+2}. We assume:

$$R_N < R_{N-2} \qquad \text{(N odd)}$$

But:

$$R_N = I + [B^{-1} + (I+R_{N-2})^{-1}]^{-1}$$

$$R_{N+2} = I + [B^{-1} + (I+R_N)^{-1}]^{-1}$$

but from lemma (1) if $I+R_N < I+R_{N-2}$ this implies $R_{N+2} < R_N$. Since the results hold for N+1,3,5, then they hold for all N.

Q.E.D.

Lemma (5) The following statements are true:

(A.5.1) All R_N (N even) are bounded above by R_1.

(A.5.2) All R_M (M odd) are bounded below by R_o.

(A.5.3) $\lim_{N\to\infty} R_N = R_{N\infty}$

(A.5.4) $\lim_{M\to\infty} R_M = R_{M\infty}$

Proof:

To show (A.5.1) is true, i.e.

$$R_o < R_2 < R_4 < R_6 < \cdots\cdots < R_1 \qquad (A.8)$$

First it is known that all R_N (N even) satisfy:

$$R_{N+2} = I + \left[\left(\frac{S_1-I}{2}\right)^{-1} + (I+R_N)^{-1}\right]^{-1}$$

or

$$R_{N+2} - I = \left[\left(\frac{S_1-I}{2}\right)^{-1} + (I+R_N)^{-1}\right]^{-1}$$

Now use lemma 2 which implies that:

$$R_{N+2} - I < \frac{S_1 - I}{2}$$

or

$$R_{N+2} < \frac{S_1 + I}{2} = R_1 \qquad (A.9)$$

Since $R_o = I < \frac{S_1+I}{2}$ for $S_1 > I$, then (A.9) implies that $R_{N+2} < R_1$ for all N even.

Q.E.D.

To show (A.5.2) is true, i.e.

$$R_1 > R_3 > R_5 > \cdots > R_M > \cdots > R_o \quad (M \text{ odd}) \qquad (A.10)$$

Proof:

$$R_1 = \frac{S_1 + I}{2} > I = R_o \quad \text{if} \quad S_1 > I$$

Hence $R_1 > R_o$, but all R_M (M odd) satisfy:

$$R_{M+2} = I + [\left(\frac{S_1 - I}{2}\right)^{-1} + (I+R_M)^{-1}]^{-1}$$

or

$$R_{M+2} - I = [\left(\frac{S_1 - I}{2}\right)^{-1} + (I+R_M)^{-1}]^{-1} > 0$$

or $R_{M+2} > I = R_o$ for all M odd. This implies (A.5.2) is true for all M odd.

Q.E.D.

The statements (A.5.3) and (A.5.4) follow from (A.5.1), (A.5.2), *lemmas (3,4) and the theorems of* monotone operators in a Banach space [23].

Lemma (6):

$$R_{N+1} - R_N < 0 \quad \text{if N is odd}$$

$$R_{N+1} - R_N > 0 \quad \text{if N is even}$$

Proof: The following expression can be calculated:

$$R_{N+1} - R_N = [\left(\frac{S_1 - I}{2}\right)^{-1} + (I+R_{N-1})^{-1}]^{-1}$$

$$- [\left(\frac{S_1 - I}{2}\right)^{-1} + (I+R_{N-2})^{-1}]^{-1}$$

rewrite this as:

$$R_{N+1} - R_N = A_1 - A_2$$

From lemma (1) we have: $A_1 - A_2 > 0$ if $R_{N-1} > R_{N-2}$ and $A_1 - A_2 < 0$ if $R_{N-1} < R_{N-2}$ but from lemma (5) we know $R_o < R_2 < R_1$

Hence:

$$R_1 - R_o > 0 \quad ,N = 0$$

$$R_2 - R_1 < 0 \quad ,N = 1$$

By mathematical induction assume $R_{N-1} < R_{N-2}$ (for N odd), then from lemma (1), $A_1 - A_2 < 0$. Since *this is true for N=3, it holds for* all N.

Q.E.D.

Assume $R_{N-1} > R_{N-2}$ (for N even). By mathematical induction:

$$A_1 - A_2 > 0 \quad \text{by lemma (1)}$$

Since this is true for N=2, it holds for all N.

Q.E.D.

Lemma (7):

For N even $\quad \lim_{N\to\infty} R_N = R_{N\infty}$

For M odd $\quad \lim_{M\to\infty} R_M = R_{M\infty}$

and $\quad R_{N\infty} = R_{M\infty}$

Proof:

The proof is by contradiction. Assume the contrary and and

$$R_{N\infty} - R_{M\infty} = \bar{\Delta} \neq 0$$

The following expression is easily calculated:

$$\bar{\Delta} = \lim_{N\to\infty} (R_{N+1} - R_N) = \lim_{N\to\infty}[B^{-1} + (I+R_{N-1})^{-1}]^{-1}$$

$$- [B^{-1} + (I+R_{N-2})^{-1}]^{-1}$$

If N is odd, then from lemma (6) this implies $\bar{\Delta} < 0$. If N is even, then from lemma (6) this implies $\bar{\Delta} > 0$. These last two statements yield a contradiction if $\bar{\Delta} \neq 0$. *Note if* $\bar{\Delta} = 0$, then everything is consistent. This completes the proof of theorem (2).

Appendix B

To show the best estimate of a matrix square root is is:

$$\hat{S}_o = \max_N \; (1/2) [R_N + R_{N-1}] \qquad (B.1)$$

Define the best estimate $\hat{S}_o$ as that estimate which minimizes:

$$\min_{\hat{S}_o} \; J = \| \hat{S}_o - S_1^{1/2} \| \qquad (B.2)$$

But from lemma (3) and (4) of appendix A we have (for some N):

$$R_o < R_2 < R_4 < \cdots < R_{2N} < S_1^{1/2} < R_{2N+1} < \cdots < R_3 < R_1 \qquad (B.3)$$

Now substituting $\hat{S}_o$ from (B.1) into (B.2) yields:

$$J = \left\| \tfrac{1}{2} (R_N - S_1^{1/2}) + \tfrac{1}{2} (R_{N-1} - S_1^{1/2}) \right\|$$

The following inequality now results:

$$J \leq \left\| \tfrac{1}{2} (R_N - S_1^{1/2}) \right\| + \left\| \tfrac{1}{2} (R_{N-1} - S_1^{1/2}) \right\| \qquad (B.4)$$

The inequality (B.4) has the smallest upper bound if R_N and R_{N-1} are chosen as the largest values of the computed apriori bounds in (B.3). The term best estimate is the guess of S_0 which minimizes the right hand side of equation (B.4).

To show that the statement (3b) of theorem (3) is true, the worst case guesses of R_N and R_{N+1} are substituted into equation (B.4). This yields the upper bound (38b).

Q.E.D.

Appendix C

One would expect to obtain bounds on Riccati equations when formulated in the factorization framework. See, e.g. [24,25] for simplification of Riccati equations when studied within the context of optimal control and estimation theory. The proof of theorem (4) now follows as previously demonstrated in Appendices A and B using the same technique. To outline the proof of theorem (4) the following lemmas can be easily shown to be true by following the similar proofs in Appendices A and B.

Lemma (8): The matrix subsequences $\bar{R}_N$ for N=0,2,4,6,$\cdots$ (all even integers) form a monotonically increasing matrix sequence, i.e.

$$\bar{R}_0 < \bar{R}_2 < \bar{R}_4 < \bar{R}_6 < \cdots < \bar{R}_N < \cdots <$$

Lemma (9): The matrix subsequences $\bar{R}_M$ for M=1,3,5,$\cdots$ (all odd integers) form a monotonically decreasing matrix sequence, i.e.

$$\bar{R}_1 > \bar{R}_3 > \bar{R}_5 > \bar{R}_7 > \cdots >$$

Lemma (10): The following statements are true:

(C.10.1) All $\bar{R}_N$ (N even) are bounded above by $\bar{R}_1$.

(C.10.2) All $\bar{R}_M$ (M odd) are bounded below by $\bar{R}_0$.

(C.10.3) $\lim_{N\to\infty} \bar{R}_N = \bar{R}_{N\infty}$

(C.10.4) $\lim_{M\to\infty} \bar{R}_M = \bar{R}_{M\infty}$

Lemma (11): $\bar{R}_{N+1} - \bar{R}_N < 0$ if N is odd

$\bar{R}_{N+1} - \bar{R}_N > 0$ if N is even

Lemma (12): For N even $\lim_{N\to\infty} \bar{R}_N = \bar{R}_{N\infty}$

For M odd $\lim_{M\to\infty} \bar{R}_M = \bar{R}_{M\infty}$

and $\bar{R}_{M\infty} = \bar{R}_{N\infty}$. All statements in theorem (4) follow when lemmas (8-12) are proved.

References

[1] Bierman,G.J.,"Sequential Square Root Filtering and Smoothing of Discrete Linear Systems",Automatica, Vol. 10, pp. 147-158, March, 1974.

[2] Bierman, G.J.,"Measurement Updating Using The U-D Factorization", Proceedings of The 1975 IEEE Conference on Decision and Control, pp. 337-346.

[3] M. Morf and T. Kailath, "Square-Root Algorithms For Least Square Estimation", IEEE Transactions on Automatic Control,Vol. AC-20, pp. 487-498, Aug.,1975.

[4] Bierman, G.J. and C.L. Thornton, "Numerical Comparison of Discrete Kalman Filter Algorithms: Orbit Determination Case Studies", 1976 IEEE Conference on Decision and Control, Clearwater, Florida, Dec.,1976.

[5] Repperger, D.W.,"A Square Root of A Matrix Approach To Obtain The Solution To A Steady State Matrix Riccati Equation", IEEE Trans. Automat. Contr., Vol. AC-21, No. 5, October, 1976, pp. 786-787.

[6] D.L. Kleinman,"On An Iterative Technique For Riccati Equation Computations", IEEE Trans. Automat. Contr., Vol. AC-13, pp. 114-115, February, 1968.

[7] D.L. Kleinman,"Iterative Solution of Algebratic Riccati Equations",IEEE Trans. Automat. Contr.,Vol. AC-19, pp. 252-254, June, 1974.

[8] D.G. Lainiotis,"Discrete Riccati Equation Solutions:Partitioned Algorithms", IEEE Trans. Automat. Contr., August, 1975, pp. 555-556.

[9] D.G. Lainiotis,"Generalized Chandrasekhar Algorithms:Time Varying Models", IEEE Trans. Automat. Contr., October, 1976, pp. 728-732.

[10] D.G. Lainiotis, "Partitioned Riccati Solutions and Integration-Free Doubling Algorithms",IEEE Trans. Automat. Contr, Vol. AC-21, No. 5, October, 1976.

[11] E.D. Denman, A. N. Beavers,"The Matrix Sign Function and Computations in Systems", Applied Mathematics and Computations, Vol. 2, pp. 63-94, 1976.

[12] A.N. Beavers and E.D. Denman,"A New Solution Method For Quadratic Matrix Equations", Mathematical Biosciences, Vol. 20, pp. 135-143, 1974.

[13] R.S. Bucy,"Global Theory of The Riccati Equation", Journal of Computer and System Sciences, Vol.1, 1967.

[14] J.E. Potter,"Matrix Quadratic Solutions",J. SIAM Applied Mathematics, Vol. 14, No. 3, May, 1966

[15] M.E. Womble, and J.E. Potter,"A Prefiltering Version of The Kalman Filter With New Numerical Integration Formulas For Riccati Equations", 1973 IEEE Conference on Decision and Control, pp63-67.

[16] C. Visser,"Notes on Linear Operators", Proc. Acad. Sci. Amsterdam, Vol. 40,pp. 270-272, 1937.

[17] R.E. Bellman,"Introduction to Matrix Analysis", McGraw-Hill, 1960, pp. 177.

[18] K.J. Astrom,"Introduction To Stochastic Control Theory", Academic Press, 1970, pp. 116-125.

[19] Hardy, G.H. and Wright, E.M.,"An Introduction To The Theory of Numbers", Oxford University Press, 1960.

[20] Niven, I. and Zucherman,"An Introduction To The Theory of Numbers", John Wiley and Sons, 1966.

[21] R.E. Kalman,"Toward a Theory of Difficulity of Computation in Optimal Control", Proceedings of The IBM Scientific Computing Symposium-Control Theory and Applications, 1966.

[22] Repperger, D.W.,"The Most Efficient Method To Numerically Compute The Solution of The Scalar Steady State Riccati Equation", 1976 International Conference on Information Sciences and Systems, Patras, Greece.

[23] L. V. Kantorovich and C.P. Akilov,"Functional Analysis in Normed Spaces", New York:MacMillan, 1964, pp. 189-190.

[24] Repperger, D.W.,"An Algorithm For Computation of The Minimum Energy Solution of Systems With Retardation", International Journal of Control, Vol. 19, No. 6, 1974, pp 1047-1055.

[25] Repperger, D.W. and A.J. Koivo,"On Stable, Forward Filtering and Fixed-Lag Smoothing in a Class of Systems With Time Delays", IEEE Transactions on Automatic Control, Vol. AC-19, pp. 266-268, June, 1974.